Vulnerability Analysis of Linux Distributions and Docker Container Images

University of Oulu
Faculty of Information Technology
and Electrical Engineering / M3S
Bachelor's Thesis
Niina Oikarinen
9.6.2021

Abstract

Docker containers are an increasingly popular alternative for virtual machines, and they are widely used in small-scale and large-scale organizations alike. Containers are usually based on Linux distributions and vulnerabilities in these distributions affect all applications built upon these containers. The purpose of this study was to analyse the current security state of selected Linux distributions and provide insight about the overall security of Docker container usage.

The goal of this study was to recognize what components and component versions were used in different OS distributions and how vulnerable these components were. The amounts and severities of vulnerabilities were compared between different OS distributions. Changes in critical and high severity vulnerabilities were compared between container distribution versions. The lifetimes and types of fixed critical and high severity vulnerabilities were determined. Along with Docker containers corresponding ISO distributions were analysed for comparison.

Analysis of ISO and container distributions of Linux-based Debian, Ubuntu, and CentOS were conducted with Black Duck Binary Analysis (BDBA) software. BDBA is used to analyse the binary code of the distributions. Analysis results contain information about identified components, their versions, and vulnerabilities associated with them.

As a result, Debian, Ubuntu, and CentOS container distributions were considered secure. The observed container maintenance strategies differed between distributions: Debian and Ubuntu containers were updated periodically (approximately monthly), whereas CentOS container updates were tied to Linux ISO image updates - i.e., official releases. The number of critical vulnerabilities were low on all lately released containers. Fixed vulnerabilities between container releases varied a lot in age and severity. Even though containers are based on ISO distributions, different versions of same components were used in them making their vulnerability profile potentially different. In all distributions, software rotting was observed, and it is suggested that only latest versions of maintained distributions should be used, if there is no specific reason to not do so.

Keywords
Software security, Linux distributions, Docker containers, Vulnerabilities, CVE

Supervisor
Adjunct Professor, Docent Raija Halonen

Foreword

This thesis was done with the support from my family and friends – you have been great help and good distraction; your support has been invaluable. I would like to thank Synopsys for employing me and offering the tools used in this thesis. I would also like to thank all my colleagues at Synopsys for welcoming me as a part of the team and teaching me, and especially Dr. Bogdan Mihaila – your guidance and expertise has been essential from the beginning to the end.

Niina Oikarinen

Oulu, January 29, 2021

Abbreviations

BDBA	Black Duck Binary Analysis
BoM	Bill of material
CVE	Common vulnerabilities and exposures
CVSS	Common vulnerability scoring system
CWE	Common weakness enumeration
CNA	CVE numbering authority
FIRST	Forum of Incident Response and Security Teams
FOSS	Free and open-source software
OS	Operating system
OSS	Open-source software
OSSRA	Open-source software risk analysis
RHEL	Red Hat Enterprise Linux
RQ	Research question
SCA	Software composition analysis

Contents

1. Introduction

The purpose of this study was to give introduction to software security and cybersecurity field and terminology, and to analyse the current security state of selected Linux distributions and provide insight about the overall security of Docker container usage. Motivation for this topic was that Linux distributions and Docker containers are widely utilized in products and services ranging from e-commerce to medical devices and security issues of a single Linux distribution version can affect all software that relies on it.

Prior research has been done to examine the security state of Linux packages and Docker containers as well as a variety of open-source software. Even though the results vary depending on the analysed software, used analysis methods, and the time of research, all lead to the same conclusion – Docker containers are affected by serious security issues. All the prior research focused on analysing large amounts of software. Even though some research focused on analysing only the latest container releases, not much data was available about the security state of actively maintained software.

Research question of this study was what kind of vulnerabilities are identified in Linux distributions. Research methods consisted of literature review and quantitative research where data was gathered by analyzing Linux distribution versions and sets of observations were formed for comparison and critical analysis purposes.

Main contribution of this study is the introduction to software security field and terminology provided with literature review and the overview to the current security state of selected Linux distributions provided with software composition analysis done with Black Duck Binary Analysis tool.

Structure of the thesis is formed as follows. After introduction, the prior research related to this topic and the used research methods are presented in Prior research and Research methods parts, respectively. These are followed by the literature review and the research itself in the Study part. After the research is introduced, the results of made analyses are presented in the Findings part. Then the findings are compared to results of prior research and observations about the findings are made in the Discussion and implications part. After discussion, the main observations are presented in the Conclusions part. At the end of this thesis are reference list. Gathered and discussed data is presented in tables that are in Appendix A.

2. Prior research

Large scale security vulnerability studies have been done to Docker images before with different analysis tools, such as Clair (Shu et al., 2017), Anchore (Liu et al., 2020) and Trivy (Prevasio, 2020). Shu et al. found that images contained more than 180 vulnerabilities on average and more than 70 vulnerabilities on average for latest-tagged official images. More than 80 % of images have at least one high severity level vulnerability. It was also notable, that latest-tagged official images were updated in less than 14 days, but when looking at all the analysed images, 50 % had not been updated in previous 200 days and 30 % had not been updated in previous 400 days.

Previous study of Linux package image security vulnerabilities is available (Falk & Henriksson, 2017), where 70 % of the images have at least one high severity level vulnerability and 54 % of the images have at least one critical severity level vulnerability. Authors also noted that official images can be not updated in several weeks or even months – and for public images this time can be years.

Recently a study of over 2 million analysed Docker images was done by Liu et al. (2020) where 30 % of analysed latest-tagged official images were found to contain at least one critical or high severity level vulnerability. For community (non-official) images the corresponding number was 64 %, indicating that official images are better maintained than community images, but both suffer serious software vulnerabilities. In the same study the vital time of 334 CVE's were analysed, and it was concluded that on average it takes 181 days for a vulnerability to be fixed in a common software - and additional 422 days (totalling to 603 days) for a vulnerability to be fixed in a Docker image.

Recently the first analysis of all the 4 million container images available in Docker Hub was conducted by Prevasio (2020). Study found that 51 % of the analysed Linux container images had at least one critical vulnerability and 13 % had at least one high level vulnerability. 20 % of images were non-vulnerable. Same study discovered 6433 malicious or possibly harmful images – that corresponds to 0,16 % of analysed Docker Hub images. While the percentage may look trivial, it possesses a viable threat as amounts of Docker Hub images and image downloads have been on a rise and Prevasio has estimated that Docker Hubs annual image downloads should top 100 billion on 2020.

Tak et al. (2018) discovered that more than 87 % of container images were based on Linux based distributions Debian, Alpine and Ubuntu. Fourth most popular operating system (OS) distribution was Linux based CentOS. Of the analysed 6589 Debian, Ubuntu, or CentOS images, 8 % contained no known vulnerabilities and on average image had 10,3 vulnerabilities while the median was 7 vulnerabilities per image. It was notable that different distributions had different vulnerability patterns: on average Ubuntu had 10,7 vulnerabilities (maximum 72 vulnerabilities), whereas Debian had 7,9 (maximum 24) and CentOS 18,5 (maximum 116) vulnerabilities. While CentOS had most vulnerabilities, it also had most non-vulnerable images (30 %). This was hypothesised to be due to CentOS having longest time since last version update (during their study), giving time for vulnerabilities to accumulate while new ones are being discovered, but also having some well-cared images that are constantly updated.

Tak et al. (2018) also identified the most vulnerable packages for three Linux based distributions, Debian, Ubuntu, and CentOS, at the time. For Debian, the top 3 was openssl, perl and sensible-utils, for Ubuntu libssl1.0.0, patch and libffi 6, and for CentOS libstdc++, libgcc and yum.

In recent Open Source Security and Risk Analysis report (OSSRA report, Synopsys, 2020b) 75 % of audited applications codebases contained vulnerabilities and 49 % of codebases contained high severity level vulnerabilities. It was also found that 91 % of codebases had components more than four years out of date or had no development activity in the last two years. Study was focused on codebases, not container images.

Martin et al. (2018) have reviewed the state of container security ecosystem based on vulnerability analyses done by other researchers. Martin et al. present typical use-cases where detected vulnerabilities can be exploited and possible ways to mitigate vulnerabilities.

3. Research methods

This study consisted of literature review and research parts. In the research part terms and phenomena introduced in literature part were observed in actual analysis done to open-source software. Utilized research methods were quantitative as gathered data was based on reliably repeatable measurements, such as amount of found critical vulnerabilities in specific Linux distribution version and sets of observations were formed for comparison and critical analysis purposes.

The purpose of this study was to examine what kind of vulnerabilities are identified in Linux distributions. This research question could be further divided to following questions.

RQ1: How the number of vulnerabilities change between Docker container versions?

RQ2: What types of vulnerabilities are fixed between versions?

RQ3: How vulnerabilities differ between desktop/server distribution versions and of the same Linux distribution?

For this research, different Linux distribution versions had to be identified and downloaded before analyzing them with BDBA software. BDBA analysis resulted in a complete software bill of materials (BoM) – a list of third-party and open-source components with identified vulnerabilities associated with these components. Further data analysis of found vulnerabilities was done to compare the amount and severity of vulnerabilities between versions.

BDBA analysis results were used to construct lists of components and vulnerabilities that are compared between versions.

Limitations

This Bachelor's thesis was limited by the depth and width of the analysed Docker images: only Linux distribution was selected for this analysis and not all versions of it can be analysed due to the sheer number of available versions and limited time reserved for this work. For this work, the focus was on the lately released Linux-based Debian, Ubuntu, and CentOS distributions. Additionally, only part of the found vulnerabilities were chosen for further examination – giving focus only to critical vulnerabilities. Found vulnerabilities were limited to vulnerabilities found with BDBA software. My plan was to try to find and remove false positives reported by the tool as far as possible using publicly available data.

4. The study

The literature review part provides an overview to software security of Linux-based open-source operating systems, followed by research part. First in the literature part free and open-source software is defined and their differences to common software products are introduced. Then, software security is explained with brief introduction to vulnerabilities, software analysis methods, and Black Duck Binary Analysis tool used in this thesis for analysing the security of Linux-based open-source operating systems. Lastly, a brief introduction for Linux-based open-source operating systems is provided.

4.1 Free and open-source software

Free and open-source software (FOSS) refers to software that is free to use, meaning that users are freely licenced to use, copy, study, change, improve and distribute the software, and its source code is openly shared. FOSS products may, or may not, be non-commercial as the "free" in FOSS refers to freedom of the product, not the price, cost, or charge. (GNU, 2019.)

FOSS enables building software products from existing packages or modules ("components"). Building new software from FOSS, or open-source software (OSS), components may decrease software development costs while increasing software security and stability as anyone can participate in component development, inspect the source code, and help find bugs and design flaws from the code. FOSS and OSS components are usually protected by licences which possible restrictions users must be aware of. (Opensource, 2021.)

4.2 Software security and cybersecurity

Software security can be defined as how well software is protected against malicious attacks and other threats while ensuring that the software continues to work correctly. Software security risks include bugs, buffer overflows, design flaws, malicious intruders, hackers, and improper digital handling. (Cyber Security Kings, 2020.)

Software security and cybersecurity are partly overlapping terms and main difference between them is that software security can be seen to be connected to a specific device whereas cybersecurity is connected to whole internet or big data (Cyber Security Kings, 2020). Most critical cybersecurity risks for web applications are called OWASP top 10 and in 2017 they were injection attack, broken authentication, sensitive data exposure, XML external entities, broken access control, security misconfiguration, cross-site scripting XSS, insecure deserialization, using components with known vulnerabilities and insufficient logging and monitoring (OWASP, 2017). All security risks must be considered early in the software design and development to achieve a secure software. Writing secure software can be ensured either by secure programming – taking security aspects continuously into account while designing and writing program – or by late detection – doing security analyses to written program and fixing found security issues. (Sampaio & Garcia, 2016.)

One way to look at cybersecurity issues is to categorize them into Five Hard Problems of cybersecurity research: scalability and composability, policy-governed secure collaboration, security-metrics-driven evaluation, design, development and deployment, resilient architectures and understanding, and accounting for human behaviour. Combining these categories to the probability of security breach – how, where and when attack will occur, and if it does, how it can be detected – a risk value for each vulnerability can be calculated. (Scala et al., 2019.)

Open-source software does not elude any OWASP top 10 cybersecurity risks, but in open-source projects, or any project recycling components, the risk of using components with known vulnerabilities should be specifically addressed. Being aware of possible known vulnerabilities affecting used components, users should first know what components and component versions their software contains. Knowing exactly what your software contains is not something to be taken as granted as projects are often developed by multiple users, not all is well documented, and used components may depend on other components that users may not consider when thinking of components comprising the developed software. (OWASP, 2021.)

For comprehensive cybersecurity risk assessment, evaluating cybersecurity risks related to each individual component is not enough. According to scalability and composability problem of Five Hard Problems, the security of system itself, not its separate components, should always be top priority of cybersecurity assessment. If the whole system is not addressed in cybersecurity risk assessment, it is possible to mitigate security risks of certain components and simultaneously shift the threat to another component, or fix vulnerabilities of components, but simultaneously modifications of components may cause new vulnerabilities to the system. (Scala et al., 2019.)

4.2.1 Vulnerabilities

Vulnerabilities are weaknesses, bugs, or design flaws, that compromise software security and can be exploited or triggered by threats (Computer Security Resource Center, 2020). Threats are intentional or accidental negative events that result in unwanted impact to a software or a system. Threat actors, such as hackers, intentionally trigger threat actions, but threats can be also triggered by insiders, authorized users, and natural disasters (Tunggal, 2020). Multiple software security tools are available both commercially and non-commercially that can be used to detect possible software security vulnerabilities affecting the analysed software products (OWASP, 2021).

CVE

Common Vulnerabilities and Exposures (CVE) is a list of common identifiers for publicly known cybersecurity vulnerabilities. CVEs, or CVE entries, are used as an industry standard for identifying unique software or firmware vulnerabilities. Using CVEs replaced company and tool specific security vulnerability databases and enabled discussion and sharing information about vulnerabilities, a baseline for tool evaluation, and automated data exchange as each vulnerability can be identified independently from the organization or tool used. (MITRE, 2020.)

Identified vulnerabilities are given a unique CVE ID number in format CVE-year-number sequence, for example, CVE-2020-28052, by CVE Numbering Authorities (CNAs). Every CVE also includes description of the vulnerability and at least one public reference. CVE is an international community effort, while being sponsored by

multiple federal organizations of USA, and free for public download and use. (MITRE, 2020.)

CVSS

Common vulnerability scoring system (CVSS) is the current standard used to evaluate the severity and exploitability of vulnerabilities. Severity assets how much damage the vulnerability can cause to software or organization whereas exploitability assets how easy or probable it is to exploit the vulnerability. CVSS is governed by Forum of Incident Response and Security Teams (FIRST). Currently used version is 3.1 – although version 2 is also seen frequently as older vulnerabilities do not have the newer version 3.1 score. (Bhatt et al., 2020.)

Both CVSS versions, 2 and 3.1, have several metrics that are used to calculate CVSS score that ranges from 0 to 10. Metrics can be divided into three categories: base metrics, temporal metrics, and environmental metrics. From these metrics, the base metrics are universal metrics whereas temporal and environmental metrics may be software or organization specific. As metrics between CVSS versions 2 and 3.1 are partly different, CVSS version 2 and version 3.1 scores cannot be compared directly. (Bhatt et al., 2020.)

Base metrics determine the severity of the vulnerability and is used by National Vulnerability Database (NVD) to rank vulnerabilities to severity categories. These categories are low (0.0 – 3.9), medium (4.0 – 6.9), and high (7.0 – 10.0) for CVSS version 2 scores and none (0.0), low (0.1 – 3.9), medium (4.0 – 6.9), high (7.0 – 8.9), and critical (9.0 – 10.0) for CVSS version 3.1 scores. Base score can be combined with temporal score to take into consideration availability of mitigations and environmental score to consider how widespread vulnerable systems are within an organization. (NVD, 2020.)

Vulnerability management

Vulnerability management is a continuous process which lasts software's whole lifecycle. Usually, the goal is to find out as many as possible vulnerabilities before releasing the software to customers. This can be achieved multiple ways, but most common is to test software before releasing it. Used tests are software and organization dependent. (Cavalancia, 2020.) Testing can be done to code itself, for example, with automated static analysis methods, or to the whole software, for example, with penetration testing (Austin et al., 2013).

Vulnerability management is usually started within the organization, before releasing the product to the customers, but is then continued within and outside of the developing organization as new vulnerabilities are continuously discovered. Some testing methods, such as penetration testing, require specific expertise that most software developers and testers does not have, and these methods are commonly outsourced to other organizations. (Austin et al., 2013.) Software products that are commonly targeted by threat actors can also participate in bug-bounty programs, where external security researchers, also known as ethical hackers or white-hat hackers, try to find vulnerabilities from the software in exchange to monetary payments and public recognition (Zhao et al., 2017).

After releasing it is also common for the software users, customer organizations, or ethical hackers to find bugs and vulnerabilities from the software and report them to the

software developing organization. Some open-source projects have even outsourced the whole bug hunting to users and other developers in the form of open bug reporting and tracking systems. (Zhao et al., 2017.)

Software rotting

As vulnerability discovery continues the whole lifetime of software, the older the software the more vulnerabilities it has accumulated if it has not been updated with newer versions or security patches. This phenomenon, where software has more and more security issues when code and software stay untouched, is known as software rotting. Software does not have more security issues as it gets older, but we are more aware of the security issues that it has always had. In some cases, changes in the environment where software is used may also expose software to more vulnerabilities or reveal vulnerabilities that were not identifiable in the original environment. (Georgescu, 2020.)

4.2.2 Software analysis methods

Software analysis methods can be divided into two groups according to the timing of the analysis: static analysis is done to source code or compiled and executable code without executing it and dynamic analysis is done to program while it executes. Both method types include a variety of different types of techniques used, for example, static software analysis tools can help to visualize code, check for correctness (such as compilers doing type checking), or recognize inefficient or insecure parts of code. Similarly, dynamic software analysis tools can, for example, measure performance, identify bugs, or visualize execution. (Nethercote, 2004.)

Other way to categorize software analysis methods is to divide them to source analysis methods and binary analysis methods. Source analysis methods use source-code for analysing software, whereas binary analysis methods use machine code, with or without statically linked object code, for analysis. Both method types can be utilized with static and dynamic analysis techniques. (Nethercote, 2004.)

Almost all software analysis methods can be done both manually and automatically. Some techniques, such as type checking, are greatly enhanced by automating the process as manual utilization is very time consuming and is prone to human errors. Adversely, some techniques do not benefit greatly from automatization as they require a lot of upkeeping due to changes in software, or are prone to computer errors, for example, in cases where test executions are not deterministic. (Garousi & Mäntylä, 2016.)

Some techniques, such as penetration testing in which cyberattacks are simulated to identify possible security vulnerabilities are compatible with both manual and automated testing. Utilizing different approaches may result in better overall coverage for testing, as automated penetration tests are good for finding implementation bugs whereas manual penetration testing is good for identifying design flaws. (Austin et al., 2013.)

It has been argued that static analysis tools for automated code review are the most effective in identifying vulnerabilities in software (McGraw, 2008). However, when comparing automated static analysis with different types of penetration testing methods, it was noticed that different techniques recognized different types of vulnerabilities. Thus, relying to only one testing method or technique can leave many vulnerabilities

unnoticed. As all vulnerabilities should be removed from a system, regardless their nature or severity, it is advisable to combine multiple testing methods and techniques to identify vulnerabilities of the system. (Austin et al. 2013.)

4.2.3 Black Duck Binary Analysis (BDBA)

Black Duck Binary Analysis (BDBA) is a commercial software composition analysis (SCA) tool by Synopsys that provides crucial information about associated licenses, security vulnerabilities, and code quality risks. Unlike many of its competitors, BDBA analyses binary code, instead of source code, which enables scanning of virtually any software, if open source, commercial applications, firmware binaries and any software for which the source code is not publicly available. (Synopsys, 2020a.)

BDBA, previously known as Protecode Supply Chain (Protecode SC) and Codenomicon Appcheck, uses a signature-based detection method to detect third-party libraries and their versions from software binaries and reports which licences and known vulnerabilities are associated with identified libraries. BDBA works on compiled binaries and supports multiple different programming languages, for example C/C++, C#, Java, and Go. In theory BDBA supports all compiled programming languages that are distributed as binary executables, but in practice formatting signatures for new programming languages requires a significant amount of research and development resources. For this reason, only the most common programming languages and executable formats are supported. (Kuuva, 2018.)

BDBA fetches vulnerability data from NVD, utilizes CVEs to identify vulnerabilities, and classifies CVEs according to CVSS scores. BDBA utilizes automated static software analysis for binary code. (Kuuva, 2018.)

4.3 Linux

Linux is a common name for a family of Linux kernel based open-source operating systems. Linux-based systems are modular Unix-like operating systems that are usually packaged in Linux distributions. Linux distributions may be commercial, such as Red Hat Enterprise Linux (RHEL) and SUSE, or non-commercial, such as Debian, Ubuntu, Fedora and CentOS. Many Linux based distributions are community efforts and, being open-source software, they may share significant parts of their source-code. For example, Ubuntu is derived from Debian and CentOS is derived from RHEL. (Canonical, 2020; CentOS, 2020.)

4.3.1 ISO images

ISO image contains all the data from an optical disk, such as CD or DVD. As ISO images contains all the information required for the program, such as whole operating system, to assemble in a single file, ISO images are often used to distribute large programs via the internet. ISO files need to be opened and assembled to be able to use the programs that are transferred. ISO files can be also mounted to virtual machines so that they can be run like CD or DVD disks. (Fisher, 2021.) Linux ISO images are installation files that contain the whole desktop or server version of Linux distribution (LinuxLookup, 2021).

4.3.2 Containers

Containers are a lightweight alternative for virtual machines. Both containers and virtual machines are used to emulate physical computers and are based on computer architectures. Their main difference is that whereas virtual machines all require their own copy of operating system (OS), containers may share the same OS kernel. When comparing virtual machines and containers, the lightweight containers have much smaller start-up times and require less resources for each image. (Kaur et al., 2017.) Containers are considered as standardized method for microservices deployment, used also by well-known large-scale companies such as Amazon, Spotify, Netflix, and Twitter (Soldani et al., 2018), but container security remains the main concern and adoption barrier for many companies (Sultan et al., 2019).

Docker images

Docker platform is widely used in software development as it provides fast and lightweight, container-based platform for developing, running, and shipping applications. It allows developers to work in standardized environments using local containers. These containers are based on Docker images – easily sharable, read-only templates with instructions for creating Docker containers. (Docker, 2020.) Docker Hub is used to share Docker images provided by software vendors, open-source projects, and community. One of the most popular Docker images are Linux distributions, such as Debian which has more than 500 million downloads on Docker Hub. (Docker Hub, 2020.)

Docker images contain almost solely Linux based OS distributions as over 97 % of top 10000 public container images from Docker Hub contain Linux based OS (Tak et al. (2018). Vulnerabilities in Docker images usually affect all applications built using these images (Mohallel et al., 2016) and thus gaining information about the current security state of Linux distribution provides some insight about overall security of Docker image usage.

4.4 Current research

Latest Debian, Ubuntu, and CentOS Linux distribution versions from years 2016 – 2020 were downloaded from the official distribution sites manually. Amd64 DVD versions of available download files were selected for Debian and Ubuntu distributions. For CentOS, version corresponding to Amd64 DVD is x86_64, thus x86_64 was selected for CentOS distribution. Downloaded distribution versions are listed in Appendix A, Table 1.

First available official releases and latest Debian, Ubuntu, and CentOS Docker official container build versions from years 2016 – 2020 were downloaded from DockerHub via terminal and saved as Docker images for analysis.

```
docker pull name:tag
```

```
docker save -o name name:tag
```

One Ubuntu container version from year 2015 was also included in the study as it still was under security maintenance. Downloaded Docker containers are listed in Appendix A, Table 2.

All container versions for year 2020 were downloaded and saved for selected distribution versions of CentOS (version 8), Debian (Buster) and Ubuntu (Bionic Beaver, Xenial Xerus). Selection criteria was that versions should be actively maintained. For Ubuntu versions, which had multiple maintained distributions unlike CentOS and Debian, Bionic Beaver and Xenial Xerus were selected as they were actively maintained and released before the beginning of 2020 – this dropped out Trusty Tahr, as it had reached the end of standard support, and both Groovy Gorilla and Focal Fossa, as they were released during 2020. Found vulnerabilities were counted for analysis date and container release date. For known vulnerabilities present during container release date, found vulnerabilities were filtered with vulnerability publishing date and all vulnerabilities that were published after container release were removed.

Collected distribution and Docker container versions were analysed with BDBA software (scanner version 20201211) during two different sessions (16.12.2020 and 14.1.2021). Due to new vulnerabilities being added to BDBA database continuously, newer BDBA analysis could have more matching vulnerabilities and vulnerable components. For each data table, vulnerability publishing dates were filtered according to the earliest analysis date used for data in that table, to ensure comparable data within data table. Due to this, care should be taken when comparing results from different tables.

All vulnerability results are presented with NVD provided CVSS version 3.1 scores. Some vulnerabilities, such as CVE-2020-29362, may have been updated between analysis sessions and may cause minor errors in the data. CVE-2020-29362 CVSS 3.1 scoring was updated from 9.1 to 5.3 and noticed when comparing critical vulnerabilities between container releases. CVSS 3.1 scoring of CVE-2020-29362 was fixed by hand to match updated information in results.

5. Findings

This chapter presents the findings of the research part of this thesis. Findings are divided to age of the analysed distribution versions, identified components, and vulnerabilities associated with identified components.

5.1 Update time

Analysed container distribution versions and their age during the analysis is presented in Appendix A, Table 2. For all the analysed latest containers, the average age was 300 days. For maintained distributions, the average age of latest containers was 56 days (100 days if security maintenance was included). In comparison, for unmaintained distributions the average age of latest containers was 477 days. It should be noted that the age of latest containers during analysis varied a lot – from 29 days of Debian Buster to 1261 days of Ubuntu Yakkety. For Debian and Ubuntu container releases the release dates are included in the container tag.

All container releases for year 2020 was examined for maintained versions of CentOS (version 8), Debian (Buster) and Ubuntu (Bionic and Xenial). Results are presented in Appendix A, Table 2. Average time between new container releases were 163 days for CentOS, 24 days for Debian, 31 days for Ubuntu Bionic, and 29 days for Ubuntu Xenial.

5.2 Found components

Amounts of found components and vulnerable components from analysed container distributions were collected from the BDBA analysis results. Results are presented in Appendix A, Table 3. Average percentage of vulnerable components in all analysed containers were 22,9 % for Debian, 20,9 % for Ubuntu, and 23,7 % for CentOS. On average in the latest container releases there were 19,2 % vulnerable components in Debian containers, 19,5 % in Ubuntu containers, and 18,9 % in the latest CentOS container. In maintained Ubuntu distribution containers, the percentage was 15,2 %. Top 3 most vulnerable components are presented in Appendix A, Table 4. Vulnerability of component was determined by adding up found vulnerabilities (#vulns) from each latest distribution included in analysis. For example, the sum of vulnerabilities associated with berkeleydb was calculated adding up berkeleydb-associated vulnerabilities of latest Bullseye, Buster, Stretch, and Jessie containers. Number of found instances of the same component does not affect the sum of vulnerabilities. Number of identified versions does affect the sum of vulnerabilities – for example, in Appendix A, Table 4, total of 76 vulnerabilities from 4 different berkeleydb versions were found from latest Debian containers and total of 209 vulnerabilities from 11 different berkeleydb versions were found from latest Ubuntu containers.

For ISO distributions corresponding results are presented in Appendix A, Table 5. To be able to compare container vulnerable components with ISO vulnerable components, few distributions were selected for component matching analysis. ISO analysis results containing only components matching components found from containers are presented in Appendix A, Table 6.

Average percentage of all vulnerable components in ISO files were 10,3 % for Debian, 14,8 % for Ubuntu (12,8 % for desktop distributions and 16,8 % for server distributions), and 22,3 % for CentOS. Analysis of only components matching container distributions resulted to 28,2 % vulnerable components in Debian Buster, 18,0 % in Ubuntu distributions (17,9 % in Groovy, 21,2 % in Bionic, and 14,9 % in Xenial), and 16,9 % in CentOS 8.3.2011. Top 3 most vulnerable components in ISO distributions are presented in Appendix A, Table 7 and top 3 most vulnerable components that can be found from ISO distributions and containers alike are presented in Appendix A, Table 8.

5.3 Found vulnerabilities

Found vulnerabilities are examined according to the amounts of vulnerabilities affecting different component versions, lifetime of current critical or high severity level vulnerabilities, and accumulation of vulnerabilities as the distribution versions age. For the amounts of vulnerabilities, container and ISO distribution vulnerabilities are examined separately.

5.3.1 Amount of container vulnerabilities

Amounts of different severity vulnerabilities were collected for the analysed container distributions and results are presented in Appendix A, Table 9. Found vulnerabilities are grouped by CVSS version 3.1 scores. Critical vulnerabilities were found in 86 % of all analysed Debian containers, 62 % of all Ubuntu containers, and 100 % all CentOS containers. All analysed containers contained high severity level vulnerabilities. It should be noted that CentOS 8 containers contain approximately double the number of components than Debian Buster or Ubuntu Bionic and Xenial containers.

Maintained latest container versions of Ubuntu had no containers with critical vulnerabilities whereas maintained latest container versions of Debian and CentOS both had critical vulnerabilities. Unreleased Debian container version (testing, currently Bullseye) had no critical vulnerabilities. Ubuntu container version with security maintenance (Trusty) had multiple critical vulnerabilities.

On average latest Debian container versions had 60,5 vulnerabilities, of which 4,5 was critical vulnerabilities and 34,8 were high severity vulnerabilities. In comparison earlier Debian container versions (first available container versions of released distribution versions) had 111,7 vulnerabilities, of which 17,0 critical vulnerabilities and 51,3 high severity level vulnerabilities.

For Ubuntu, the latest versions had on average 65,2 vulnerabilities, of which 6,5 were critical and 37,2 were high severity level vulnerabilities. Earlier Ubuntu versions had 92,3 vulnerabilities, of which 11,6 were critical and 47,8 were high severity level vulnerabilities.

5.3.2 Amount of ISO distribution vulnerabilities

For ISO distributions corresponding results are presented in Appendix A, Table 10. Similarly to component results, selected vulnerability results are also matched to contain only components found in containers for better comparison. These results are shown in Appendix A, Table 11. All analysed ISO distributions were ‘latest’ available releases at the time of the study and were matched to the ‘latest’ available containers while downloading ISO distributions.

While looking at the total vulnerabilities found in ISO distributions, analysed Debian ISO distributions had an average of 2515,5 vulnerabilities, of which on average 199,3 were critical and 890,0 were high severity level vulnerabilities. Analysed Ubuntu ISO distributions had an average of 2001,5 vulnerabilities (desktop distribution 1958,0 vulnerabilities and server distribution 2044,9 vulnerabilities), of which 150,9 were critical and 713,9 were high severity level vulnerabilities. Similarly, analysed CentOS 8.3.2011 ISO distribution had 354 vulnerabilities, of which 26 were critical and 130 were high severity level vulnerabilities.

For the matching vulnerabilities between ISO distributions and containers, the number of vulnerabilities found in Debian Buster was 54, of which 1 were critical and 38 high severity level during analysis, and in CentOS 8.3.2011 there was 83 vulnerabilities, of which 5 were critical and 46 were high severity level vulnerabilities during analysis. Average amount of vulnerabilities in Ubuntu distributions during analysis was 37,5, of which Groovy had 1 critical vulnerability, whereas Bionic and Xenial had 0 critical vulnerabilities, and on average they all had 26,5 high severity level vulnerabilities.

5.3.3 Lifetime of vulnerabilities

Lifetime of vulnerabilities was tracked in container releases of Ubuntu Buster, Ubuntu Bionic and Xenial, and CentOS 8 during year 2020. Results of analysis are presented in Appendix A, Table 12. Lifetime of a vulnerability was calculated subtracting the publishing date of vulnerability from the date of container version release in which the vulnerability is fixed.

Goal was to identify critical vulnerabilities that were fixed during this time and calculate the time from finding of the vulnerability to fixing it, but it turned out that all critical vulnerabilities in Ubuntu containers were found after releasing the latest analysed container. Debian Buster had only one known vulnerability (CVE-2019-9893) during container releases, but it was not fixed during the observation period.

CentOS containers released during year 2020 had three critical vulnerabilities that were fixed during observation period. These vulnerabilities are presented in Appendix A, Table 13 and their average lifetime is 311 days. Additionally, two critical vulnerabilities were identified (CVE-2020-27780 and CVE-2020-29362) that were published immediately after container release where they were fixed.

Due to low number of critical vulnerabilities found and fixed during observation period, the goal was adjusted to identify critical or high severity level vulnerabilities that were fixed during observation period. Results are presented in Appendix A, Table 13. Debian Buster had 6 fixed high severity level vulnerabilities with average lifetime of 67 days. Ubuntu Bionic had 1 fixed high severity level vulnerabilities with lifetime of 646 days, Ubuntu Xenial no fixed high severity level vulnerabilities during observation period. CentOS had 18 fixed high severity level vulnerabilities with average lifetime of 419 days (minimum 195 days, maximum 898 days).

5.3.4 Software rotting

Software rotting was observed in container releases of Ubuntu Buster, Ubuntu Bionic and Xenial, and CentOS 8 for one year period (year 2020). Vulnerabilities found during container releases are presented in Appendix A, Table 12, whereas vulnerabilities found from the same containers during analysis are presented in Appendix A, Table 14.

6. Discussion and implications

The purpose of this study was to examine what kind of vulnerabilities are identified in Linux distributions. This question was further defined as (RQ1) how the number of vulnerabilities change between Docker container versions, (RQ2) what types of vulnerabilities are fixed between versions, and (RQ3) how vulnerabilities differ between different distribution versions. Additionally, collected data will be compared to previous research done to Linux distribution containers.

6.1 Change in vulnerabilities between container versions

First research question (RQ1) addressed the change in vulnerabilities between container versions. This question can be further divided to examination of how often new distribution versions are released and examination of changes in vulnerabilities between version releases. Changes in vulnerabilities between version releases can be divided to vulnerabilities that are fixed in the newer releases and vulnerabilities that are newly introduced in the newer releases.

6.1.1 Update time

According to the gathered data (Appendix A, Table 14), the availability of new container images is dependent on examined distribution: on average it was 24 days in Debian, 30 days in Ubuntu, and 163 days in CentOS. Large difference in average update times is probably caused by different approaches to container updates, as in CentOS container updates seemed to be tied to other version releases, meaning each version, such as 8.3, is released as an ISO distribution and as a container at approximately same time, whereas in Debian and Ubuntu container updates seemed to be more periodical (monthly) and not tied to point releases.

In their study, Falk and Henriksson (2017) noted that official images may not have container updates in several weeks or even months. This is very vague expression, but as different distributions have very different approaches to container updates, this can be seen as appropriate depiction of the average update times.

Average age of all analysed containers (Appendix A, Table 14) were 642 days – 318 for latest-tagged containers and 52 days for actively maintained latest-tagged containers. In their research Shu et al. (2017) concluded that most of latest-tagged official images, as also the analysed Linux distributions are, had been updated in less than 14 days, which is more often than what was observed with Debian, Ubuntu, or CentOS containers. Hence, we cannot confirm that conjecture from the data used in our study.

6.1.2 Changes in vulnerabilities

Appearance of critical and high severity level vulnerabilities were analyzed during year 2020 in maintained containers (Appendix A, Table 12). As a result, no change in critical vulnerabilities were observed in Debian Buster, Ubuntu Bionic, and Ubuntu Buster containers. In CentOS 8, three critical vulnerabilities were fixed, and one new critical

vulnerability was found. For high severity level vulnerabilities, the corresponding numbers were in Debian Buster five vulnerabilities fixed and 11 new vulnerabilities found, in Ubuntu Bionic one fixed and two new vulnerabilities found, in Ubuntu Xenial zero fixed and two new vulnerabilities found, and in CentOS 8 21 fixed and 16 new high severity level vulnerabilities found. Fixed critical and high severity level vulnerabilities are presented in Appendix A, Table 13.

It was also noted, that in CentOS 8, two critical level vulnerabilities were fixed right before vulnerabilities were published. It is assumed that publication of these vulnerabilities was intentionally postponed after release date of the fix to prevent exploitation of vulnerabilities.

Additionally, it should be mentioned that used classification of vulnerability severities in this study is based on basic metrics of CVSS 3.1 scoring. Combining base score with temporal and environmental scores gives an organization specific CVSS score, which is more accurate description of the actual severity of the vulnerability. At least RHEL, which CentOS is based on, calculates organization specific CVSS scores for vulnerabilities affecting their products – it is presumable that CentOS relies more on these RHEL based CVSS scores than the base scores received from NVD. RHEL based CVSS scores of the five critical vulnerabilities found in CentOS 8.3.2011 container classifies only one of the vulnerabilities as critical. One of these vulnerabilities is considered to be of high severity, two of medium severity, and one of low severity. If CentOS relies on these organization specific scores, it is no wonder that these vulnerabilities have not been prioritized to be fixed as soon as possible.

It can be concluded that containers with relatively short release cycle (new release once a month) had only few changes in vulnerabilities during the observation period, whereas containers with longer release cycle (release twice a year) had larger amounts of fixed and found vulnerabilities. It should be also noted that CentOS 8 containers contain approximately double the number of components than Debian Buster or Ubuntu Bionic and Xenial containers. Compared to that, Debian Buster had relatively more new vulnerabilities and relatively less fixed vulnerabilities than CentOS 8.

6.2 Fixed vulnerabilities between versions

Second research question (RQ2) concentrates on the vulnerabilities that are identified to be fixed between version releases. Firstly, how long vulnerabilities affect distributions on average is calculated. Then, the question of how the number of vulnerabilities change between container versions is addressed. Lastly, what are the types of vulnerabilities that are fixed between container versions is examined.

6.2.1 Lifetime of fixed vulnerabilities

The lifetimes of fixed vulnerabilities are presented in Appendix A, Table 13. On average, fixed critical vulnerabilities found during this study had a lifetime of 311 days and on average fixed high severity vulnerabilities had a slightly longer lifetime of 343 days. High severity level vulnerability lifetimes varied from 41 days to 898 days. In their research, Liu et al. (2020) concluded that on average it takes 603 days for a vulnerability to be fixed in container – 422 days longer than making the fix in the non-container version of the software. No such delay to fix vulnerabilities in containers was observed in our study.

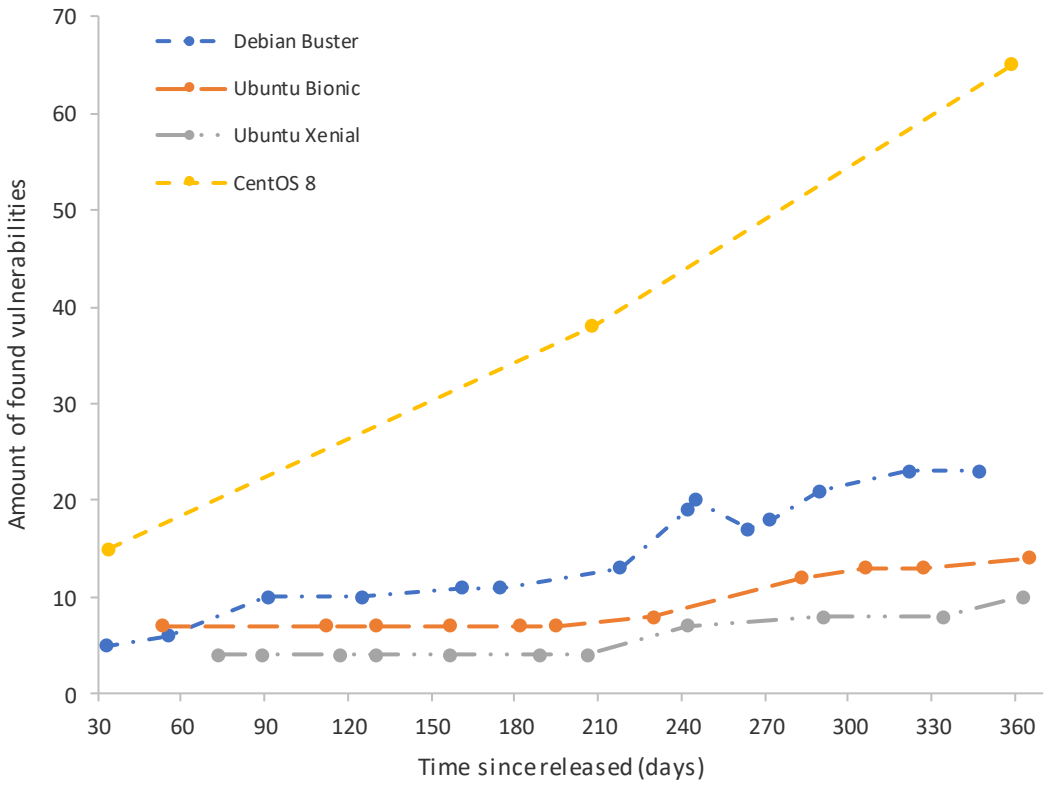
Comparing the average lifetimes, looks like critical and high severity level vulnerabilities concerning examined Linux containers (Debian Buster, Ubuntu Bionic, Ubuntu Xenial, and CentOS 8) are more promptly fixed than vulnerabilities on average. As the severity level and selection criteria of vulnerabilities analysed by Liu et al. was not presented, it is difficult to assess how much CVSS scoring or affected distributions affect the lifetime of vulnerabilities.

6.2.2 Software rotting

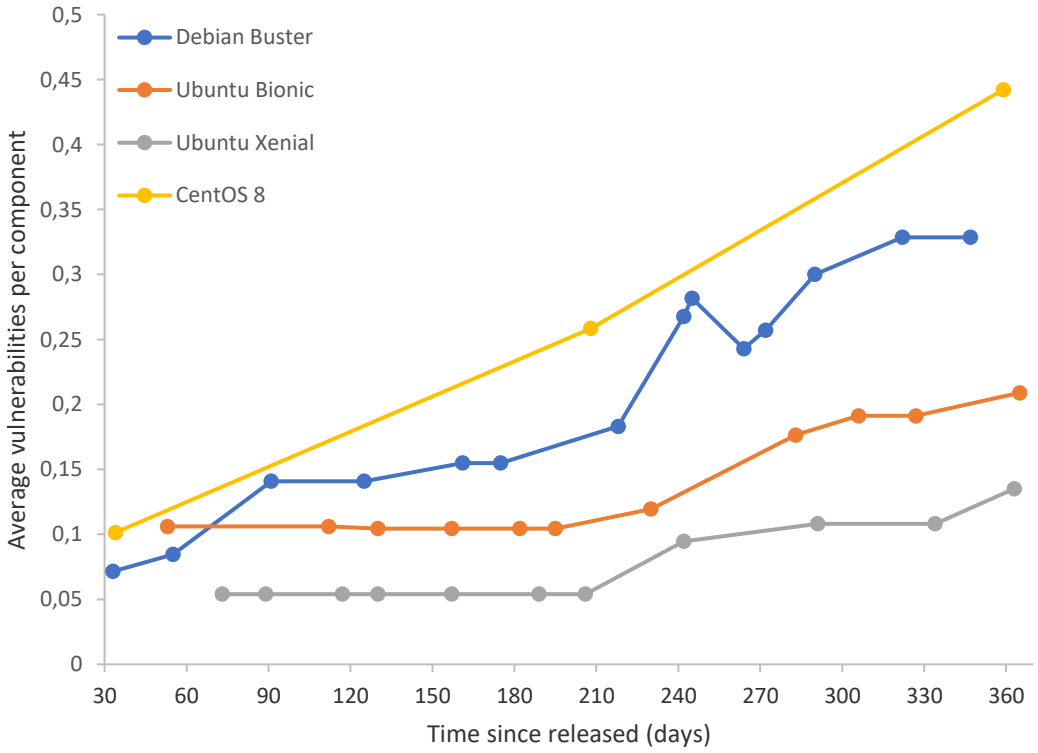
For critical and high severity level vulnerabilities the changes in vulnerabilities between container versions has been mostly answered as a part of RQ1, but for all vulnerabilities the change in vulnerabilities after container releases are presented in Figure 1a. Change in vulnerabilities after container release was calculated by subtracting numbers of vulnerabilities found during container release (Appendix A, Table 12) from vulnerabilities found during container analysis (Appendix A, Table 14). Changes in only critical vulnerabilities were not visualised as the changes were minor in all observed containers during year 2020. As the number of components is different in all analysed distributions, accumulation of vulnerabilities was normalized by dividing the amount of found vulnerabilities with the number of components. Normalized results are presented in Figure 1b.

Figure 1

a) Accumulation of Vulnerabilities as Containers Age



b) Normalized Accumulation of Vulnerabilities as Containers Age



From the visualisation in Figure 1a, it is notable that different distributions accumulate vulnerabilities at different rates. This is most probably due to different maintenance strategies of containers as the size of the distributions does not seem to play major part when comparing normalized data (number of vulnerabilities per number of components) of different distributions in Figure 1b. The trend of accumulating vulnerabilities as containers gets older, or software rotting, is observed in all analysed containers.

As Tak et al. (2018) discovered that the most popular Linux based container distributions were Debian, Alpine, Ubuntu, and CentOS, and clear majority of Linux based containers were based on them, keeping these distributions as secure as possible is of the utmost importance. Our study scope included Debian, Ubuntu, and CentOS distributions. Alpine being the current second most popular Linux based container distribution, Alpine would be substantial addition for further research. If not separately taken care of, all vulnerabilities in these containers are also inherited to containers based on these distributions. Thus, our results can directly be applied to any container using those distributions as a base layer. As software rotting is a real issue, special care should be taken to regularly update used containers to latest versions to minimise the inherited security issues.

6.2.3 Types of fixed vulnerabilities

From the 28 fixed vulnerabilities identified during year 2020 (Appendix A, Table 13), all three critical vulnerabilities and two high severity level vulnerabilities were chosen for closer inspection. As all critical vulnerabilities affected CentOS 8 container distribution, high severity level vulnerabilities were chosen so that at least one vulnerability from each distribution version were inspected. Selected vulnerabilities

with their current National Vulnerability Database (NVD) descriptions and Common Weakness Enumeration (CWE) descriptions are presented in Appendix A, Table 15.

Of the inspected vulnerabilities, three (CVE-2019-8457, CVE-2019-5482, and CVE-2019-5481) were different types of memory buffer errors, one (CVE-2020-10878) was related to improper calculation or conversion of numbers, and one (CVE-2018-17953) was access control vulnerability. Improper access control vulnerability is the only one of these listed in OWASP top 10 web application security list. Improper access control is part of OWASP broken access control category. Exploitation of vulnerabilities affecting access control can lead to attackers gaining unauthorized access to software as a common user or administrator. This may lead to compromised data if attackers can create, access, update, or delete data as they wish. (OWASP, 2017)

6.3 RQ3: Differences between distribution versions

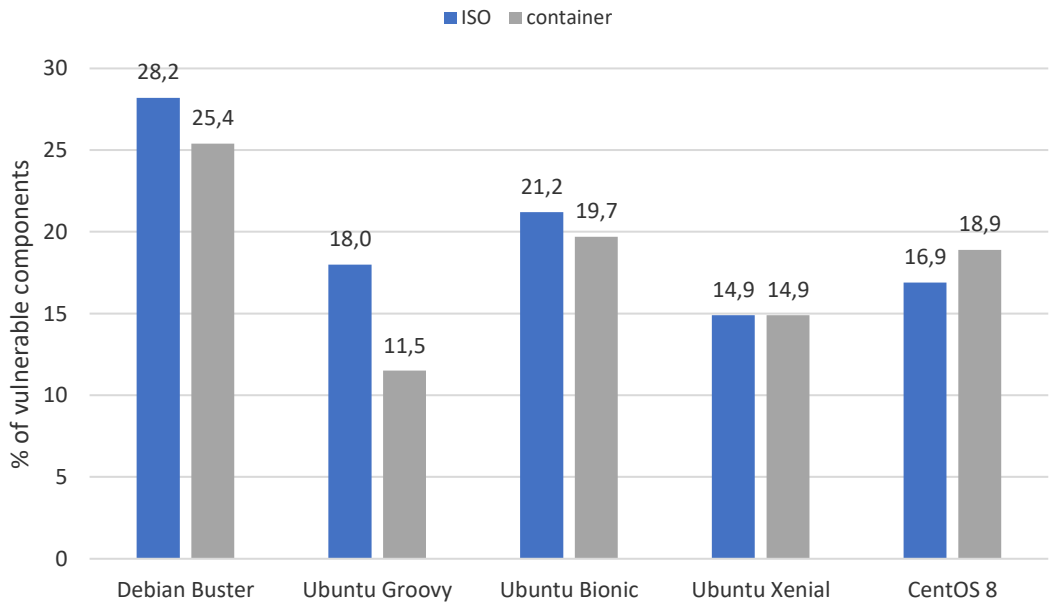
Last of the research questions (RQ3) addresses differences between different distribution versions, namely ISO distributions, that can be further divided to desktop and server versions in Ubuntu, and container distributions. Here differences have been measured as number of vulnerable components and number of found vulnerabilities. Containers are usually constructed using a small subset of components used in original ISO distributions. As ISO distributions have considerably larger number of components than containers have, for ISO distributions both the total and container matched numbers of vulnerable components were counted. Server versions of ISO distributions have usually fewer components than desktop versions, but these components may have significant differences. As all analysed ISO distributions were latest available versions, latest versions of containers are used as comparison. It should be noted that CentOS 8 have a synced container and ISO distribution release, whereas in Debian and Ubuntu container releases are done more often than ISO distribution releases.

6.3.1 ISO distributions vs container distributions

When comparing container matching components found in ISO distributions with components found in latest-tagged containers, ISO distribution components were mostly more vulnerable than container components. Difference in vulnerable components between container and ISO distributions are presented in Figure 2.

Figure 2

Difference in Percentage of Vulnerable Components Between ISO and Container Distributions

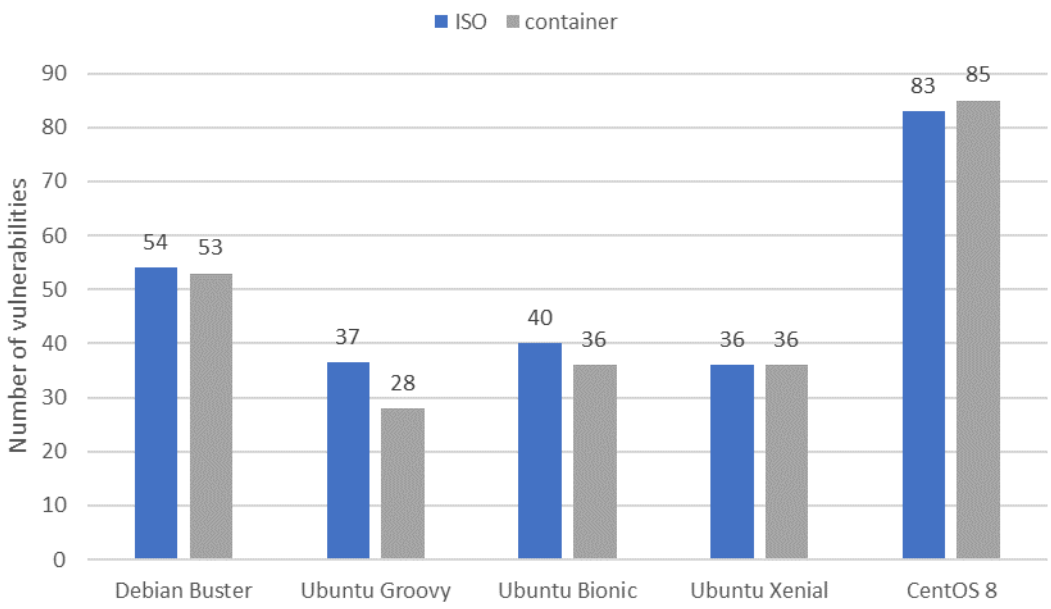


Debian and Ubuntu containers having less vulnerable components than ISO distributions could be due to analysed container releases being newer than ISO distributions and updated more often. Interestingly in CentOS, container distribution has more vulnerable components than ISO distribution. As container and ISO distribution releases are synchronized in CentOS, it is interesting that they have different number of vulnerable components, indicating that same versions of components were not used to compose different distribution versions of the same release at the same time.

Comparison of all found vulnerabilities between container distributions and container matching components of ISO distributions are presented in Figure 3. When comparing critical vulnerabilities, only Ubuntu Groovy had different number of vulnerabilities in compared distributions - zero in container distribution and one in container matching components of ISO distribution. Difference in total vulnerabilities between compared ISO and container distributions were low - all distribution versions had close to no differences in the amounts of vulnerable components. Comparing component versions in both container distributions and container matching components of ISO distributions, we observed that container matching components of ISO distributions analysis results showed more versions of the same components than container analysis results. As also the versions used by containers were present, it seems that increase in vulnerabilities and vulnerable components is due to ISO distributions using multiple versions of the components. As ISO distributions may contain thousands of components, keeping all component dependencies up to date is prone to errors.

Figure 3

Difference in Number of Vulnerabilities Between ISO and Container Distributions



Significant differences between number of vulnerabilities in compared container distributions and in container matching components of ISO distributions were observed only in Ubuntu Groovy distributions. In Ubuntu Groovy, the container matched components of ISO distribution had 32 % more vulnerabilities than the container distribution. This may be due to fact that Ubuntu Groovy was first released significantly later than other distributions - late 2020 compared to Debian Buster and CentOS 8 released originally in 2019 and other Ubuntu distributions being even older than that.

Difference of total vulnerabilities between CentOS 8 distributions and other distributions is in line with the difference of number of total components in distributions.

Liu et al. (2020) calculated that on average it took 422 days longer for a vulnerability to be fixed in a container compared to common software. In Debian, Ubuntu, or CentOS no clear delay between fixing a vulnerability in ISO distributions and container distributions was observed. This hints at distributions treating common software and container with same priority.

6.3.2 Vulnerabilities in containers compared to prior research

In prior studies the average number of found vulnerabilities vary a lot. This is probably due to different container distributions being selected for the analysis and usage of different analysis tools to gather data.

Shu et al. (2017) discovered that latest-tagged containers (not necessarily Linux-based) had on average more than 70 vulnerabilities, whereas Tak et al. (2018) calculated that analysed Debian, Ubuntu, and CentOS containers had an average of 10,3 vulnerabilities. Average amount of vulnerabilities found in all latest containers (Appendix A, Table 9) is 65,3 vulnerabilities (on average 44,2 vulnerabilities, if only maintained latest

containers are considered), which is significantly higher than the average presented by Tak et al. and close to the average presented by Shu et al. It is difficult to say whether the difference comes from the selection of containers to analyse or the analysis method itself.

In their study Tak et al. (2018) observed that different analysed distributions, Debian, Ubuntu, and CentOS, had different vulnerability patterns - on average Debian had 7,9 (maximum 24), Ubuntu had 10,7 vulnerabilities (maximum 72), and CentOS 18,5 (maximum 116) vulnerabilities. Similarly, different patterns can be seen in containers analysed with BDBA (Appendix A, Table 9) – also here Debian had lowest number of vulnerabilities (on average 60,5 for latest containers), followed closely by Ubuntu (on average 65,2 vulnerabilities), and CentOS had most vulnerabilities (85 vulnerabilities). While Tak et al. hypothesised that CentOS had most vulnerabilities due to it having longest time since container update, allowing time for software rotting, now collected analysis data refutes this analysis as CentOS was the most vulnerable of the containers but now it was also the most recent container to be added to Docker Hub.

Other often used indicator for recognizing vulnerable containers in literature was portion of containers having at least one critical or high severity level vulnerability. Falk and Henriksson (2017) reported that 54% of containers in their analysis had at least one critical vulnerability and 70 % had at least one high severity level vulnerability, whereas corresponding numbers in study by Prevasio (2020) were 51 % for critical vulnerabilities and 64 % for high severity level vulnerabilities. Both studies were done by analysing Linux containers, but not limiting the analysis to Debian, Ubuntu, and CentOS distributions. Now collected data resulted in 70 % of containers having at least one critical vulnerability and all analysed containers having high severity level vulnerabilities. Prevasio also reported that 20 % of containers were non-vulnerable – in BDBA analysis, non-vulnerable containers were not found. As with the numbers of vulnerabilities found, it is difficult to say whether the difference comes from the selection of containers to analysis or the analysis method itself – probably it is a combination of both.

In their prior research Tak et al. (2018) discovered that most vulnerable components of Debian container distributions were openssl, perl and sensible-utils. None of these components are in current Debian container top 3 vulnerable components. Openssl was now 21st and perl was 13th most vulnerable component. Sensible-utils was found from Debian containers but contained no known vulnerabilities.

Most vulnerable components of Ubuntu container distributions were libssl1.0.0, patch, and libffi 6 (Tak et al. 2018). None of these components are in current Ubuntu container top 3 vulnerable components. Libssl, also known as openssl, was now 35th most vulnerable component of Ubuntu containers. Patch was not found at all from Ubuntu containers whereas libffi was found but did not contain any known vulnerabilities.

Most vulnerable components of CentOS container distributions were libstdc++, libgcc and yum (Tak et al. 2018). None of these components are in current CentOS container top 3 vulnerable components. Libstdc++ and libgcc were both found under the name gcc. Gcc contained no known vulnerabilities. Yum has been rewritten as DNF package manager and it contains no known vulnerabilities.

6.3.3 Server distributions vs desktop distributions

Ubuntu ISO distributions had separate versions for desktop ISO distributions and server ISO distributions. Comparison of components found from each version revealed that on

average desktop versions had 596 (41,5 %) more components than server distributions (Appendix A, Table 5). However, this varied a lot as at largest the difference was 803 components and at the smallest 78 components.

Top 3 most vulnerable components of Ubuntu server distributions were linux_kernel, mysql, and imagemagick, and for Ubuntu desktop distributions the top 3 was linux_kernel, thunderbird, and binutils. In desktop distributions imagemagick was 4th most vulnerable component and mysql was 14th most vulnerable component. In server distributions thunderbird was not used at all and binutils was 8th most vulnerable component. All compared components, linux_kernel, imagemagick, mysql, and binutils, also had different number of recognized versions and found vulnerabilities – for example, in all ubuntu ISO desktop distributions 18 different versions of linux_kernel with a total of 8073 vulnerabilities were found, whereas in all ubuntu ISO server distributions 26 different linux_kernel versions with a total of 9220 vulnerabilities were found. It can be concluded that server and desktop distributions are using different versions of these components.

When comparing number of vulnerabilities in these distributions, on average server distributions had 2045 vulnerabilities (min 768, max 4894) and desktop distributions had 1958 vulnerabilities (min 676, max 3918) (Appendix A, Table 10). This was surprising as on average server distributions contain considerably less components than desktop distributions, indicating that the components and component versions used in server distributions are on average significantly more vulnerable than the components and versions used in desktop distributions. It is also possible that finding vulnerabilities in server distributions is considered more critical than finding vulnerabilities on desktop distributions and at least part of the difference is caused by more rigorous testing.

Interestingly the number of found critical vulnerabilities was smaller in server distributions, where on average 139 critical vulnerabilities were identified (min 42, max 331), than in desktop distributions, where on average 163 critical vulnerabilities were identified (min 40, max 426). Same trend can be also seen in the number of found high severity level vulnerabilities, where server distributions had 695 vulnerabilities and desktop distributions had 732 vulnerabilities on average. This suggests that even though server distributions have more total vulnerabilities on average, they are still cared relatively well compared to desktop distributions. It should be noted that these numbers contain all the latest releases of ISO distributions from year 2016 to 2020 and most of these distributions are not maintained anymore.

7. Conclusions

In this study, different ISO and container distributions of three Linux-based distributions, Debian, Ubuntu, and CentOS, are identified, downloaded, and analysed. Analysis is done with BDBA software provided by Synopsys. BDBA uses the binary code of analysed products to identify what components are used in the distributions and what vulnerabilities are affecting the identified components. Information about the vulnerable components and found vulnerabilities is collected from the BDBA analysis results and used to construct lists presented in this study. The purpose of this study is to examine what kind of vulnerabilities are identified in analysed distributions. Also, the number of vulnerable components and found vulnerabilities are compared between different distribution types as well as to prior research.

As a result, the current security status of maintained Debian, Ubuntu, and CentOS ISO and container distributions can be established and compared between distributions. As is expected due to the significant size difference of ISO and container distributions, ISO and container distributions cannot be reasonably compared. Thus, comparison is done between container distributions and the part of ISO distribution components, that are also found in comparable containers. Conclusion after comparison is that different versions of same components are used in ISO distributions and container distributions. While the number of vulnerable components differ between ISO and container distributions, no significant difference in the number or severity of found vulnerabilities is observed. Different vulnerabilities may affect comparable components of ISO and container distributions, but the overall security status is similar in both. When comparing whole ISO distributions to container distributions, maintained containers are considerably more secure than latest ISO distributions.

Analysis of maintained distributions released during year 2020 reveals that maintained Debian, Ubuntu, and CentOS containers are relatively well cared after. Debian and Ubuntu containers are updated periodically (approximately monthly), whereas CentOS container updates are tied to Linux ISO image updates - i.e., official releases. Low numbers of critical and high severity level vulnerabilities are observed in Debian and Ubuntu distributions and these can be considered very secure. CentOS has more components than other distributions and correspondingly it also has more vulnerabilities. Most of the critical vulnerabilities found in latest CentOS container are classified as non-critical by RHEL, which CentOS is based on. This suggests that also CentOS can be considered very secure as the identified critical vulnerabilities may not be applicable in CentOS products.

Comparison of maintained and un-maintained distributions confirms that un-maintained distributions, may they be ISOs or containers, are significantly more vulnerable than maintained distributions. Software rotting is observed in all analysed containers.

As a conclusion, maintained Debian and Ubuntu container distributions are more secure and more regularly updated than CentOS container distribution. No matter which distribution is selected, it is at utmost importance to keep used containers updated, always pull latest-tagged containers if there is no specific reason to not do so and rebuild containers periodically. This is especially important for widely used Linux-based distributions as vulnerabilities in these containers are usually inherited to containers build on them.

References

Austin, A., Holmgreen, C., & Williams, L. (2013). A comparison of the efficiency and effectiveness of vulnerability discovery techniques. Information and Software Technology 55 (2013) 1279–1288. https://doi.org/10.1016/j.infsof.2012.11.007

Bhatt, N., Anand, A., & Yadavalli, V.S.S. (2020). Exploitability prediction of software vulnerabilities. Quality and Reliability Engineering International. Advance online publication. https://doi.org/10.1002/qre.2754

Canonical. (2020). The Story of Ubuntu. https://ubuntu.com/about

Cavalancia, N. (2020). Vulnerability management explained. https://cybersecurity.att.com/blogs/security-essentials/vulnerability-management-explained

CentOS. (2020). CentOS Linux. https://www.centos.org/about/

Computer Security Resource Center. (2020). Vulnerability. https://csrc.nist.gov/glossary/term/vulnerability

Cyber Security Kings. (2020). Software Security vs Cyber Security: Do you Know the Difference? https://cybersecuritykings.com/2020/04/25/10-differences-between-software-security-vs-cyber-security/#:~:text=This%20may%20sound%20similar%20to,Information%20security

Docker. (2020). Docker overview. https://docs.docker.com/get-started/overview/

Docker Hub. (2020). Debian. https://hub.docker.com/_/debian

Falk, M., & Henriksson, O. (2017). *Static Vulnerability Analysis of Docker Images* (Master's thesis, Blekinge Tekniska Högskola, Karlskrona, Sweden). http://bth.diva-portal.org/smash/record.jsf?dswid=-2461

Fisher, T. (2021). What is an ISO file? https://www.lifewire.com/iso-file-2625923

Garousi, V. & Mäntylä, M. (2016). When and what to automate in software testing? A multi-vocal literature review. Information and Software Technology 76, August 2016, 92-117. https://doi.org/10.1016/j.infsof.2016.04.015

Georgescu, E. (2020). Software Rot and Cybersecurity: Why Code Degradation Is Crucial to Business Safety. https://heimdalsecurity.com/blog/software-rot/

GNU. (2019). What is free software? https://www.gnu.org/philosophy/free-sw.html

Kaur, K., Dhand, T., Kumar N. & Zeadally, S. (2017). Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers. IEEE Wireless Communications, 24 (3) 48 – 56. https://doi.org/10.1109/MWC.2017.1600427

Kuuva, O. (2018). Linux Kernel Module Recognition Using Static Binary Analysis [Master's Thesis, University of Oulu]. Jultika University of Oulu repository. http://jultika.oulu.fi/Record/nbnfioulu-201803081334

LinuxLookup. (2021). Linux ISO image downloads. https://www.linuxlookup.com/linux_iso

Liu, P., Ji, S., Fu, L., Lu, K., Zhang, X., Lee, WH., Lu, T., Chen, W. & Beyah, R. (2020). Understanding the Security Risks of Docker Hub. In Chen, L., Li, N., Liang, K. & Schneider, S. (Eds.), *Computer Security - ESORICS 2020* (pp. 257 – 276). Springer. https://doi.org/10.1007/978-3-030-58951-6

Martin, A., Raponi, S., Combe, T., & Di Pietro, R. (2018). Docker ecosystem – Vulnerability Analysis. *Computer Communications, 122*, 30–43. https://doi.org/10.1016/j.comcom.2018.03.011

MITRE. (2020). CVE Common Vulnerabilities and Exposures. https://cve.mitre.org/

MITRE. (2021). CWE Common Weakness Enumeration. http://cwe.mitre.org/data/index.html

Mohallel, A.A., Bass, J.M. & Dehghantaha, A. (2016). Experimenting with Docker: Linux Container and BaseOS Attack Surfaces. *International Conference on Information Society*. https://doi.org/10.1109/i-Society.2016.7854163

Nethercote, N. (2004). Dynamic binary analysis and instrumentation. [Doctoral dissertation, University of Cambridge] Computer Laboratory, University of Cambridge. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-606.html

NVD. (2020). Vulnerability Metrics. https://nvd.nist.gov/vuln-metrics/cvss

NVD. (2021). Vulnerability Database. https://nvd.nist.gov/vuln/search

Opensource. (2021). What is open source? https://opensource.com/resources/what-open-source

OWASP. (2017). OWASP Top Ten 2017. https://owasp.org/www-project-top-ten/2017/

OWASP. (2021). Component analysis. https://owasp.org/www-community/Component_Analysis

Prevasio. (2020). Operation Red Kangaroo: Industry's First Dynamic Analysis of 4 million Publicly Available Docker Hub Container Images. https://prevasio.com/static/web/viewer.html?file=/static/Red_Kangaroo.pdf

Sampaio, L. & Garcia, A. (2016). Exploring context-sensitive data flow analysis for early vulnerability detection. The Journal of Systems and Software 113, 337–361. https://doi.org/10.1016/j.jss.2015.12.021

Scala, N.M., Reilly, A.C., Goethals P.L. & Cukier M. (2019). Risk and the Five Hard Problems of Cybersecurity. Risk Analysis, 39, (10), 2119–2126. https://doi.org/10.1111/risa.13309

Shu, R., Gu, X., & Enck, W. (2017). A Study of Security Vulnerabilities on Docker Hub. *CODASPY '17: Seventh ACM Conference on Data and Application Security and Privacy*, 269–280. https://doi.org/10.1145/3029806.3029832

Soldani, J., Tamburri, D.A. & Van Den Heuvel, W.-J. (2018). The pains and gains of microservices: A Systematic grey literature review. Journal of Systems and Software, 146, 215-232. https://doi.org/10.1016/j.jss.2018.09.082

Sultan, S., Ahmad, I. & Dimitriou, T. (2019). Container Security: Issues, Challenges, and the Road Ahead. IEEE access, 7, 52976 – 52996. https://doi.org/10.1109/ACCESS.2019.2911732

Synopsys. (2020a). Black Duck Binary Analysis. https://www.synopsys.com/content/dam/synopsys/sig-assets/datasheets/blackduck-binaryanalysis-ds-ul.pdf

Synopsys. (2020b). Open Source Security and Risk Analysis report. https://www.synopsys.com/software-integrity/resources/analyst-reports/2020-open-source-security-risk-analysis.html

Tak, B., Kim, H., Suneja, S., Isci, C. & Kudva, P. (2018). Security Analysis of Container Images Using Cloud Analytics Framework. In: Jin H., Wang Q. & Zhang LJ. (Eds.), *Web Services – ICWS 2018* (pp. 116 – 133). https://doi.org/10.1007/978-3-319-94289-6_8

Tunggal, A.T. (2020). What is a Cyber Threat? https://www.upguard.com/blog/cyber-threat

Zhao, M., Laszka, A., & Grossklags, J. (2017). Devising Effective Policies for Bug-Bounty Platforms and Security Vulnerability Discovery. Journal of Information Policy, 7 (2017) 372-418. https://doi.org/10.5325/jinfopoli.7.2017.0372

Appendix A. Data tables

Table 1

Downloaded and Analysed ISO Distributions

Distribution release version	Alternative names	Latest release	Notes
Debian			
10.6.0	Buster, stable	26.9.2020	Maintained
9.13.0	Stretch, oldstable	18.7.2020	Unmaintained
8.11.1	Jessie, oldoldstable	23.6.2018	Unmaintained
bullseye-DI-alpha2	Bullseye, testing	16.3.2020	Not released
Ubuntu			
20.10-desktop	Groovy Gorilla	22.10.2020	Maintained
20.10-live-server			Maintained
20.04.1-desktop	Focal Fossa	6.8.2020	Maintained
20.04.1-live-server			Maintained
19.10-desktop	Eoan Ermine	17.10.2019	Unmaintained
19.10-live-server			Unmaintained
19.04-desktop	Disco Dingo	18.4.2019	Unmaintained
19.04-live-server			Unmaintained
18.10-desktop	Cosmic Cuttlefish	18.10.2018	Unmaintained
18.10-live-server			Unmaintained
18.04.5-desktop	Bionic Beaver	13.8.2020	Maintained
18.04.5-live-server			Maintained
17.10.1-desktop	Artful Aardvark	19.10.2017	Unmaintained
17.10.1-server			Unmaintained
17.04-desktop	Zesty Zapus	13.4.2017	Unmaintained
17.04-server			Unmaintained
16.10-desktop	Yakkety Yak	13.10.2016	Unmaintained
16.10-server			Unmaintained
16.04.7-desktop	Xenial Xerus	13.8.2020	Maintained
16.04.7-server			Maintained
CentOS			
8.3.2011		7.12.2020	Maintained

Table 2

Downloaded and Analysed Container Distributions

Distribution release version	Notes
Debian	
debian:buster-20201209	
debian:buster-20201117	'latest' during download
debian:buster-20201012	
debian:buster-20200908	
debian:buster-20200803	
debian:buster-20200720	
debian:buster-20200607	
debian:buster-20200514	
debian:buster-20200511	
debian:buster-20200422	
debian:buster-20200414	
debian:buster-20200327	
debian:buster-20200224	
debian:buster-20200130	
debian:buster-20190708	First official release
debian:stretch-20201117	'latest'
debian:stretch-20170620	Earliest available
debian:jessie-20201117	'latest'
debian:jessie-20170606	Earliest available
debian:bullseye-20201117	'latest'
Ubuntu	
ubuntu:groovy-20201022.1	'latest'
ubuntu:focal-20201008	'latest'
ubuntu:focal-20200423	First official release
ubuntu:eoan-20200608	'latest'
ubuntu:eoan-20191017	First official release
ubuntu:disco-20200114	'latest'
ubuntu:disco-20190423	First official release
ubuntu:cosmic-20190719	'latest'
ubuntu:cosmic-20181018	First official release
ubuntu:bionic-20201119	
ubuntu:bionic-20200921	'latest' during download
ubuntu:bionic-20200903	
ubuntu:bionic-20200807	
ubuntu:bionic-20200713	
ubuntu:bionic-20200630	
ubuntu:bionic-20200526	
ubuntu:bionic-20200403	
ubuntu:bionic-20200311	
ubuntu:bionic-20200219	
ubuntu:bionic-20200112	
ubuntu:bionic-20180426	First official release

Distribution release version	Notes
ubuntu:artful-20180706	'latest'
ubuntu:artful-20171019	First official release
ubuntu:zesty-20171122	'latest'
ubuntu:zesty-20170517.1	First official release
ubuntu:yakkety-20170704	'latest'
ubuntu:yakkety-20161013	First official release
ubuntu:xenial-20201030	
ubuntu:xenial-20201014	'latest' during download
ubuntu:xenial-20200916	
ubuntu:xenial-20200903	
ubuntu:xenial-20200807	
ubuntu:xenial-20200706	
ubuntu:xenial-20200619	
ubuntu:xenial-20200514	
ubuntu:xenial-20200326	
ubuntu:xenial-20200212	
ubuntu:xenial-20200114	
ubuntu:xenial-20160422	First official release
ubuntu:trusty-20191217	'latest'
ubuntu:trusty-20150427	First official release
CentOS	
centos:centos8.3.2011	'latest'
centos:centos8.2.2004	
centos:centos8.1.1911	Earliest available

Table 3

Found Vulnerable Components in Container Distributions

Distribution release version	Components		
	Total	Vulnerable	% vulnerable
Debian			
debian:buster-20201117	71	18	25,4
debian:buster-20190708	70	19	27,1
debian:stretch-20201117	64	15	23,4
debian:stretch-20170620	64	20	31,3
debian:jessie-20201117	87	17	19,5
debian:jessie-20170606	80	20	25,0
debian:bullseye-20201117	84	7	8,3
Ubuntu			
ubuntu:groovy-20201022.1	78	9	11,5
ubuntu:focal-20201008	72	8	11,1
ubuntu:focal-20200423	71	7	9,9
ubuntu:eoan-20200608	70	10	14,3
ubuntu:eoan-20191017	69	11	15,9
ubuntu:disco-20200114	69	15	21,7
ubuntu:disco-20190423	68	17	25,0
ubuntu:cosmic-20190719	66	15	22,7
ubuntu:cosmic-20181018	66	16	24,2
ubuntu:bionic-20200921	66	13	19,7
ubuntu:bionic-20180426	66	16	24,2
ubuntu:artful-20180706	69	18	26,1
ubuntu:artful-20171019	69	18	26,1
ubuntu:zesty-20171122	64	17	26,6
ubuntu:zesty-20170517.1	64	17	26,6
ubuntu:yakkety-20170704	64	18	28,1
ubuntu:yakkety-20161013	64	18	28,1
ubuntu:xenial-20201014	74	11	14,9
ubuntu:xenial-20160422	75	16	21,3
ubuntu:trusty-20191217	159	28	17,6
ubuntu:trusty-20150427	144	34	23,6
CentOS			
centos:centos8.3.2011	148	28	18,9
centos:centos8.1.1911	147	42	28,6

Table 4

Top 3 Most Vulnerable Components in Containers

#vulns	Debian	#vulns	Ubuntu	#vulns	CentOS
76	berkeleydb	209	berkeleydb	19	berkeleydb
56	glibc	116	glibc	17	binutils
19	systemd	60	ncurses / systemd	6	libarchive

Table 5

Found Vulnerable Components in ISO Distributions

Distribution release version	All components		
	Total	Vulnerable	% vulnerable
Debian			
10.6.0	3152	257	8,2
9.13.0	3143	339	10,8
8.11.1	2621	399	15,2
bullseye-DI-alpha2	3243	229	7,1
Ubuntu			
20.10-desktop	1535	114	7,4
20.10-live-server	698	81	11,6
20.04.1-desktop	1542	139	9,0
20.04.1-live-server	687	93	13,5
19.10-desktop	1567	189	12,1
19.10-live-server	669	114	17,0
19.04-desktop	1581	236	14,9
19.04-live-server	743	153	20,6
18.10-desktop	1494	242	16,2
18.10-live-server	732	156	21,3
18.04.5-desktop	1338	133	9,9
18.04.5-live-server	535	84	15,7
17.10.1-desktop	1228	192	15,6
17.10.1-server	1150	210	18,3
17.04-desktop	1373	219	16,0
17.04-server	1074	201	18,7
16.10-desktop	1380	231	16,7
16.10-server	1066	208	19,5
16.04.7-desktop	1320	137	10,4
16.04.7-server	1048	121	11,5
CentOS			
8.3.2011	300	67	22,3

Table 6

ISO Distribution Components That Are Also Present in Corresponding Containers

Distribution release version	Components		
	Total	Vulnerable	% vulnerable
Debian			
10.6.0	71	20	28,2
Ubuntu			
20.10-desktop	78	13	16,7
20.10-live-server	78	15	19,2
18.04.5-desktop	66	14	21,2
18.04.5-live-server	66	14	21,2
16.04.7-desktop	74	11	14,9
16.04.7-server	74	11	14,9
CentOS			
8.3.2011	148	25	16,9

Table 7

Top 3 Most Vulnerable Components in ISO Distributions

#vulns	Debian	#vulns	Ubuntu	#vulns	CentOS
1468	linux_kernel	16754	linux_kernel	175	linux_kernel
876	mysql	2280	mysql	23	binutils
759	imagemagick	1647	imagemagick	19	berkeleydb

Table 8

Top 3 Most Vulnerable Components in ISO Distributions That Are Also Present in Corresponding Containers

#vulns	Debian	#vulns	Ubuntu	#vulns	CentOS
115	glibc	994	openssl	23	binutils
95	berkeleydb	668	sqlite3	19	berkeleydb
40	zlib	663	glibc	16	openssl

Table 9

Found Vulnerabilities in Container Distributions

Distribution release version	Vulnerabilities						
	Total	Critical	High	Medium	Low	None	Missing CVSS3.1 classification
Debian							
debian:buster-20201117	53	1	37	11	4	0	0
debian:buster-20190708	59	2	40	14	3	0	0
debian:stretch-20201117	76	5	36	31	3	0	1
debian:stretch-20170620	131	23	56	47	4	0	1
debian:jessie-20201117	88	12	44	28	2	0	2
debian:jessie-20170606	145	26	58	56	3	0	2
debian:bullseye-20201117	25	0	22	2	1	0	0
Ubuntu							
ubuntu:groovy-20201022.1	28	0	24	3	1	0	0
ubuntu:focal-20201008	27	0	25	2	0	0	0
ubuntu:focal-20200423	27	0	24	3	0	0	0
ubuntu:eoan-20200608	30	0	25	5	0	0	0
ubuntu:eoan-20191017	31	0	26	5	0	0	0
ubuntu:disco-20200114	45	0	35	9	1	0	0
ubuntu:disco-20190423	49	3	35	10	1	0	0
ubuntu:cosmic-20190719	53	2	36	13	2	0	0
ubuntu:cosmic-20181018	67	7	40	16	4	0	0
ubuntu:bionic-20200921	36	0	25	9	1	0	1
ubuntu:bionic-20180426	46	5	28	11	1	0	1
ubuntu:artful-20180706	95	12	45	34	3	0	1
ubuntu:artful-20171019	97	12	46	35	3	0	1
ubuntu:zesty-20171122	126	22	57	41	4	0	2
ubuntu:zesty-20170517.1	127	22	58	41	4	0	2
ubuntu:yakkety-20170704	132	22	60	42	4	0	4
ubuntu:yakkety-20161013	133	21	60	44	4	0	4
ubuntu:xenial-20201014	36	0	25	9	1	0	1
ubuntu:xenial-20160422	51	6	29	14	1	0	1
ubuntu:trusty-20191217	109	13	52	33	2	0	9
ubuntu:trusty-20150427	295	40	132	92	7	0	24
CentOS							
centos:centos8.3.2011	85	5	46	33	0	0	1
centos:centos8.1.1911	139	8	71	56	3	0	1

Table 10

Found Vulnerabilities in ISO Distributions

Distribution release version	Vulnerabilities						
	Total	Critical	High	Medium	Low	None	Missing CVSS3.1 classification
Debian							
10.6.0	1481	92	425	621	96	0	247
9.13.0	2516	190	946	1212	84	0	84
8.11.1	4485	432	1763	1942	136	0	212
bullseye-DI-alpha2	1580	83	426	705	137	0	229
Ubuntu							
20.10-desktop	816	44	318	365	22	0	67
20.10-live-server	768	42	296	349	25	0	56
20.04.1-desktop	1110	53	419	522	28	0	88
20.04.1-live-server	935	50	348	439	38	0	60
19.10-desktop	1685	87	598	873	58	0	69
19.10-live-server	1341	78	493	659	52	0	59
19.04-desktop	1875	147	686	875	60	0	107
19.04-live-server	1077	102	421	453	38	0	63
18.10-desktop	1757	184	632	811	74	0	56
18.10-live-server	1171	118	456	512	48	0	37
18.04.5-desktop	676	40	249	291	35	0	61
18.04.5-live-server	771	49	297	360	26	0	39
17.10.1-desktop	2801	222	1001	1445	89	0	44
17.10.1-server	4894	276	1561	2793	220	0	44
17-04-desktop	3599	341	1419	1694	94	0	51
17.04-server	3701	257	1205	2023	178	0	38
16.10-desktop	3918	426	1487	1795	93	0	117
16.10-server	4141	331	1285	2228	187	0	110
16.04.7-desktop	1343	89	515	645	47	0	47
16.04.7-server	1650	82	592	862	68	0	46
CentOS							
8.3.2011	354	26	130	178	7	0	13

Table 11

Found Container Matching Vulnerabilities in ISO Distributions

Distribution release version	Vulnerabilities						
	Total	Critical	High	Medium	Low	None	Missing CVSS3.1 classification
Debian							
10.6.0	54	1	38	11	3	0	1
Ubuntu							
20.10-desktop	35	1	27	5	1	0	1
20.10-live-server	38	1	28	7	1	0	1
18.04.5-desktop	40	0	27	10	1	0	2
18.04.5-live-server	40	0	27	10	1	0	2
16.04.7-desktop	36	0	25	9	1	0	1
16.04.7-server	36	0	25	9	1	0	1
CentOS							
8.3.2011	83	5	46	30	1	0	1

Table 12

Found Vulnerabilities in Maintained Containers During Container Releases

Distribution release version	Vulnerabilities during container release						
	Found	Critical	High	Medium	Low	None	Missing CVSS3.1 classification
Debian 10							
debian:buster-20201209	47	1	35	9	2	0	0
debian:buster-20201117	47	1	35	9	2	0	0
debian:buster-20201012	47	1	35	9	2	0	0
debian:buster-20200908	47	1	35	9	2	0	0
debian:buster-20200803	46	1	34	9	2	0	0
debian:buster-20200720	54	1	39	12	2	0	0
debian:buster-20200607	52	1	38	11	2	0	0
debian:buster-20200514	42	1	33	6	2	0	0
debian:buster-20200511	42	1	33	6	2	0	0
debian:buster-20200422	42	1	33	6	2	0	0
debian:buster-20200414	41	1	32	6	2	0	0
debian:buster-20200327	39	1	30	6	2	0	0
debian:buster-20200224	37	1	29	5	2	0	0
debian:buster-20200130	37	1	29	5	2	0	0
Ubuntu 18.04							
ubuntu:bionic-20201119	34	0	25	8	0	1	0
ubuntu:bionic-20200921	34	0	25	8	0	1	0
ubuntu:bionic-20200903	34	0	25	8	0	1	0
ubuntu:bionic-20200807	35	0	26	8	0	1	0

Distribution release version	Vulnerabilities during container release						
	Found	Critical	High	Medium	Low	None	Missing CVSS3.1 classification
ubuntu:bionic-20200713	35	0	26	8	0	1	0
ubuntu:bionic-20200630	35	0	26	8	0	1	0
ubuntu:bionic-20200526	34	0	25	8	0	1	0
ubuntu:bionic-20200403	33	0	25	7	0	1	0
ubuntu:bionic-20200311	32	0	24	7	0	1	0
ubuntu:bionic-20200219	32	0	24	7	0	1	0
ubuntu:bionic-20200112	32	0	24	7	0	1	0
Ubuntu 16.04							
ubuntu:xenial-20201030	34	0	25	8	0	1	0
ubuntu:xenial-20201014	34	0	25	8	0	1	0
ubuntu:xenial-20200916	34	0	25	8	0	1	0
ubuntu:xenial-20200903	35	0	25	9	0	1	0
ubuntu:xenial-20200807	35	0	25	9	0	1	0
ubuntu:xenial-20200706	35	0	25	9	0	1	0
ubuntu:xenial-20200619	35	0	25	9	0	1	0
ubuntu:xenial-20200514	32	0	24	7	0	1	0
ubuntu:xenial-20200326	31	0	23	7	0	1	0
ubuntu:xenial-20200212	31	0	23	7	0	1	0
ubuntu:xenial-20200114	31	0	23	7	0	1	0
CentOS 8							
centos:centos8.3.2011	75	5	43	26	0	1	0
centos:centos8.2.2004	87	4	48	33	1	1	0
centos:centos8.1.1911	86	7	46	30	2	1	0

Table 13

Lifetimes of Vulnerabilities

CVE	CVSS3	Lifetime (days)
Debian Buster		
CVE-2020-10878	8.6	59
CVE-2020-10543	8.2	59
CVE-2020-1712	7.8	41
CVE-2020-12723	7.5	59
CVE-2020-13777	7.4	60
CVE-2020-11501	7.4	122
Ubuntu Bionic		
CVE-2018-17953	8.1	646
CentOS 8		
CVE-2019-8457	9.8	384
CVE-2019-5482	9.8	275
CVE-2019-5481	9.8	275
CVE-2018-1000858	8.8	719
CVE-2019-5018	8.1	578
CVE-2019-3844	7.8	418
CVE-2019-3843	7.8	418
CVE-2019-5436	7.8	386
CVE-2020-13110	7.8	206
CVE-2018-14404	7.5	699
CVE-2019-15847	7.5	289
CVE-2019-16056	7.5	285
CVE-2019-20454	7.5	298
CVE-2019-19956	7.5	350
CVE-2019-19906	7.5	355
CVE-2019-15903	7.5	461
CVE-2019-13050	7.5	528
CVE-2018-20843	7.5	898
CVE-2020-13630	7.0	195
CVE-2020-1752	7.0	222
CVE-2020-1751	7.0	235

Table 14

Found Vulnerabilities in Maintained Containers During BDBA Analysis

Distribution release version	Age during analysis (days)	Vulnerabilities during container analysis						
		Found	Critical	High	Medium	Low	None	Missing CVSS3.1 classification
Debian 10								
debian:buster-20201209	33	52	1	38	10	3	0	0
debian:buster-20201117	55	53	1	38	11	3	0	0
debian:buster-20201012	91	57	1	40	12	4	0	0
debian:buster-20200908	125	57	1	40	12	4	0	0
debian:buster-20200803	161	57	1	40	12	4	0	0
debian:buster-20200720	175	65	1	45	15	4	0	0
debian:buster-20200607	218	65	1	45	15	4	0	0
debian:buster-20200514	242	61	1	44	12	4	0	0
debian:buster-20200511	245	62	1	44	13	4	0	0
debian:buster-20200422	264	59	1	43	12	3	0	0
debian:buster-20200414	272	59	1	43	12	3	0	0
debian:buster-20200327	290	60	1	44	12	3	0	0
debian:buster-20200224	322	60	1	44	12	3	0	0
debian:buster-20200130	347	60	1	44	12	3	0	0
Ubuntu 18.4								
ubuntu:bionic-20201119	53	41	1	28	10	1	0	1
ubuntu:bionic-20200921	112	41	1	28	10	1	0	1
ubuntu:bionic-20200903	130	41	1	28	10	1	0	1
ubuntu:bionic-20200807	157	42	1	29	10	1	0	1
ubuntu:bionic-20200713	182	42	1	29	10	1	0	1
ubuntu:bionic-20200630	195	42	1	29	10	1	0	1
ubuntu:bionic-20200526	230	42	1	29	10	1	0	1
ubuntu:bionic-20200403	283	45	1	29	12	1	0	2
ubuntu:bionic-20200311	306	45	1	29	12	1	0	2
ubuntu:bionic-20200219	327	45	1	29	12	1	0	2
ubuntu:bionic-20200112	365	46	1	30	12	1	0	2
Ubuntu 16.4								
ubuntu:xenial-20201030	73	38	1	26	9	1	0	1
ubuntu:xenial-20201014	89	38	1	26	9	1	0	1
ubuntu:xenial-20200916	117	38	1	26	9	1	0	1
ubuntu:xenial-20200903	130	39	1	26	10	1	0	1
ubuntu:xenial-20200807	157	39	1	26	10	1	0	1
ubuntu:xenial-20200706	189	39	1	26	10	1	0	1
ubuntu:xenial-20200619	206	39	1	26	10	1	0	1
ubuntu:xenial-20200514	242	39	1	26	10	1	0	1
ubuntu:xenial-20200326	291	39	1	26	10	1	0	1
ubuntu:xenial-20200212	334	39	1	26	10	1	0	1
ubuntu:xenial-20200114	363	41	1	27	11	1	0	1

Distribution release version	Age during analysis (days)	Vulnerabilities during container analysis						
		Found	Critical	High	Medium	Low	None	Missing CVSS3.1 classification
CentOS 8								
centos:centos8.3.2011	34	90	5	50	33	1	0	1
centos:centos8.2.2004	208	98	6	48	41	2	0	1
centos:centos8.1.1911	359	151	8	73	65	4	0	1

Table 15

Types of Inspected Vulnerabilities

CVE	CVSS3	NVD current description [a]	CWE description [b]
CVE-2020-10878	8.6	Perl before 5.30.3 has an integer overflow related to mishandling of a "PL_regkind[OP(n)] == NOTHING" situation. A crafted regular expression could lead to malformed bytecode with a possibility of instruction injection.	CWE-190: The software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the calculation is used for resource management or execution control.
CVE-2019-8457	9.8	SQLite3 from 3.6.0 to and including 3.27.2 is vulnerable to heap out-of-bound read in the rtreenode() function when handling invalid rtree tables.	CWE-125: The software reads data past the end, or before the beginning, of the intended buffer.
CVE-2019-5482	9.8	Heap buffer overflow in the TFTP protocol handler in cURL 7.19.4 to 7.65.3.	CWE-120: The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.
CVE-2019-5481	9.8	Double-free vulnerability in the FTP-kerberos code in cURL 7.52.0 to 7.65.3.	CWE-415: The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.
CVE-2018-17953	8.1	A incorrect variable in a SUSE specific patch for pam_access rule matching in PAM 1.3.0 in openSUSE Leap 15.0 and SUSE Linux Enterprise 15 could lead to pam_access rules not being applied (fail open).	CWE-284: The software does not restrict or incorrectly restricts access to a resource from an unauthorized actor.

Note. [a] NVD (2021), [b] MITRE (2021)